i

OCI Flexvolumes Driver HowTo

Oracle Container Services for use with Kubernetes® (OCSK) provides a certified version of Kubernetes to users of Oracle Linux. To further enhance the offering and continue integration with the Oracle Cloud Infrastructure (OCI), the Oracle Linux team are providing access to pre-built RPMs containing the OCI Flexvolume Driver. Fully tested on OCSK version 1.1.9 this technical preview of the OCI Flexvolume Driver packages is available from the Oracle Linux yum server and Oracle ULN developer channels.

Although Kubernetes already provides support for multiple volume options, Flexvolumes were introduced in the Kubernetes 1.8 release to enable users to write their own drivers and add support for their own volumes. Oracle provides a Flexvolume driver for Kubernetes clusters running on Oracle Cloud Infrastructure (OCI). The driver facilitates mounting OCI block storage volumes to Kubernetes Pods via the Flexvolume plugin interface.

The Oracle Cloud Infrastructure Block Volume service lets you dynamically provision and manage block storage volumes. You can create, attach, connect and move volumes as needed to meet your storage and application requirements. Once attached and connected to an instance, you can use a volume like a regular hard drive. Volumes can also be disconnected and attached to another instance without the loss of data.

Pre-requisite: The instructions below assume the user has already configured a Block Volume in Oracle Cloud. For information on how to create a Block Volume please refer to the documentation. In addition, if you OCI networking behind a firewall, then you must add a proxy variable to your kube-controller-manager:

Modify `/etc/kubernetes/manifests/kube-controller-manager.yaml` and add the following env variable (OCI_PROXY):

Example:

```
name: kube-controller-manager
env:
 - name: OCI_PROXY
   value: http://www-proxy.org:8080
```

Install / Setup

Follow the instructions here to set up Oracle Container Services for use with Kubernetes (OCSK) in your OCI environment, make sure to pay particular attention to the section about "Requirements to Use Oracle Container Services for use with Kubernetes on Oracle Cloud Infrastructure"

The OCI Flexvolume driver binary must be installed on every node in your Kubernetes cluster. The OCI Flexvolume driver (oci-flexvolume-driver-0.6.2-2.0.2.el7.x86_64.rpm) is available in the Oracle Linux 7 developer channel on Oracle Yum and Unbreakable Enterprise Network (ULN).

If your environment uses the Oracle Linux yum servers, you must enable the ol7_developer repository on each node in the cluster. For example, you can run the following command on each node:

```
# yum-config-manager --enable ol7_developer
```

Alternatively, edit the `/etc/yum.repos.d/public-yum-ol7.repo`

```
[ol7_developer]
name=Oracle Linux $releasever Development Packages ($basearch)
baseurl=https://yum.oracle.com/repo/OracleLinux/OL7/developer/$basearch/
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-oracle
gpgcheck=1
enabled=1
```

If your environment uses the Unbreakable Linux Network (ULN), you must first subscribe your systems to the `ol7_x86_64_developer` channel on each node in the cluster. For example, you can run the following command on each node:

```
# uln-channel -a -c ol7_x86_64_developer
```

Alternately:

1. Log in to http://linux.oracle.com with your ULN user name and password.
2. On the Systems tab, click the link named for the system in the list of registered machines.
3. On the System Details page, click **Manage Subscriptions**.
4. On the System Summary page, select each required channel from the list of available channels and click the right arrow to move the channel to the list of subscribed channels.

   Subscribe the system to the `ol7_x86_64_developer` channel.

5. Click **Save Subscriptions**.

Now install the package

```
# yum install oci-flexvolume-driver
```

The driver is installed in the volume plugin path on every node in your Kubernetes cluster at the following location:`/usr/libexec/kubernetes/kubelet-plugins/volume/exec/oracle~oci/oci`.

NOTE: If running kube-controller-managers in a container you *must* ensure that the plugin directory is mounted into the container.

Configuration

The driver requires API credentials for a OCI account with the ability to attach and detach OCI block storage volumes from to/from the appropriate nodes in the cluster. For more information please refer to this page.

Provide these credentials in a YAML file on the master nodes in the cluster at `/usr/libexec/kubernetes/kubelet-plugins/volume/exec/oracle~oci/config.yaml` in the following format:

```
---
auth:
  tenancy: <tenancy>
  compartment: <compartment>
  user: <user>
  region: <region>
  key: |
 -----BEGIN RSA PRIVATE KEY-----
 <snip>
 -----END RSA PRIVATE KEY-----
  passphrase: <passphrase>
  fingerprint: 11:22:33:44:55:66:77:88:99:10
  vcn: <vcn>
```

Copy the config.yaml file to the remaining nodes in the cluster (every node in the cluster will use this config file). Make sure the file is placed in the same location on all clusters: /usr/libexec/kubernetes/kubelet-plugins/volume/exec/oracle~oci/config.yaml

<note: if using Kubernetes version 1.10 (OCSK 1.1.10) - this is the final step and you can proceed to the Example section>

Make Flexvolume plugin available in kube-controller-manager (this step is required if using Kubernetes 1.9 (OCSK 1.1.9)

Modify `/etc/kubernetes/manifests/kube-controller-manager.yaml`

1. Add the following under volumeMounts:

```
- mountPath: /usr/libexec/kubernetes/kubelet-plugins/volume/exec
   name: flexvolume-dir
```

Example:

```
      volumeMounts:
        - mountPath: /etc/kubernetes/pki
          name: k8s-certs
          readOnly: true
        - mountPath: /etc/ssl/certs
          name: ca-certs
          readOnly: true
        - mountPath: /etc/kubernetes/controller-manager.conf
          name: kubeconfig
          readOnly: true
        - mountPath: /usr/libexec/kubernetes/kubelet-plugins/volume/exec
          name: flexvolume-dir
        - mountPath: /etc/pki
          name: ca-certs-etc-pki
          readOnly: true
```

2. Add the following hostPath statement under volumes:

```
- hostPath:
   path: /usr/libexec/kubernetes/kubelet-plugins/volume/exec
   type: DirectoryOrCreate
 name: flexvolume-dir
```

Example :

```
     volumes:
      - hostPath:
          path: /etc/kubernetes/pki
          type: DirectoryOrCreate
        name: k8s-certs
      - hostPath:
          path: /etc/ssl/certs
          type: DirectoryOrCreate
        name: ca-certs
      - hostPath:
          path: /etc/kubernetes/controller-manager.conf
          type: FileOrCreate
        name: kubeconfig
      - hostPath:
          path: /usr/libexec/kubernetes/kubelet-plugins/volume/exec
          type: DirectoryOrCreate
        name: flexvolume-dir
      - hostPath:
          path: /etc/pki
          type: DirectoryOrCreate
```

3. Restart the kubeadm service by issuing the following command on the master node:

```
kubeadm-setup.sh restart
```

4. The OCI Flexvolume driver setup is complete. The following sample will walk through how to create a pod using the block volume. This requires the OCID for the target block volume.

Note The OCID for block volumes can be found by logging into your OCI console, selecting Menu → Block Storage → Block Volumes and selecting the Volume name.

Example: Creating a pod hosting a Nginx service using an OCI volume

Step 1. Create a nginx.yaml file using the following

```
[root@kubemaster system]# cat /root/flexvol/config/nginx.yaml
apiVersion: v1
kind: Pod
metadata:
  name: flexnginx
  labels:
    app: flexnginx
spec:
  containers:
  - name: nginx
    image: nginx
    ports:
    - containerPort: 80
    volumeMounts:
    - name: "abuwcljrnuyj67r7xgtfryj6yqps47tgze3k753f5ljlj5pzvbpsfb3f5ibq"
      mountPath: /usr/share/nginx/html
  #nodeSelector:
   # node.info/availability.domain: 'OBfV-US-ASHBURN-AD-2'
  volumes:
  # The volume name must be the last section of the OCID of the volume
being
  # attached (. separated). e.g. if the volume ocid was
  #
"ocid1.volume.oc1.phx.aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaa"
  # the volume name would be
"aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"
  - name: "abuwcljrnuyj67r7xgtfryj6yqps47tgze3k753f5ljlj5pzvbpsfb3f5ibq"
    flexVolume:
      driver: "oracle/oci"
      fsType: "ext4"
```

NOTE 1: Here the name `"abuwcljrnuyj67r7xgtfryj6yqps47tgze3k753f5ljlj5pzvbpsfb3f5ibq"` is the last '.' separated section of the volume OCID
NOTE 2: Make sure to create your block volume in the same availability domain as your nodes

Step 2. Create the pods

```
## Create pod
[root@kubemaster config]# kubectl create -f nginx.yaml
```

```
pod "flexnginx" created

## Check if pod is running or not
[root@kubemaster config]# kubectl  get po -l app=flexnginx
NAME        READY     STATUS    RESTARTS   AGE
flexnginx   1/1    Running   0          11m

## Check pod details
[root@kubemaster config]# kubectl  describe po  flexnginx
Name:         flexnginx
Namespace:    default
Node:         kubeworker2/10.0.1.128
Start Time:   Thu, 12 Apr 2018 06:38:55 +0000
Labels:       app=flexnginx
Annotations:  <none>
Status:       Running
IP:           10.244.2.36
Containers:
  nginx:
    Container
ID:   docker://f5892e1084febf44b3fde81ae663f3495f1378e84b287958271a174
683bc846c
    Image:          nginx
    Image ID:       docker-
pullable://nginx@sha256:37350fbb4afbb1c01b6e542fe1537dd701e4430983d6d9
c673cbb5eccdbec357
    Port:           80/TCP
    State:          Running
      Started:      Thu, 12 Apr 2018 06:39:15 +0000
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /usr/share/nginx/html from
abuwcljrnuyj67r7xgtfryj6yqps47tgze3k753f5ljlj5pzvbpsfb3f5ibq (rw)
      /var/run/secrets/kubernetes.io/serviceaccount from default-
token-xkljq (ro)
Conditions:
  Type           Status
  Initialized    True
  Ready          True
  PodScheduled   True
Volumes:
  abuwcljrnuyj67r7xgtfryj6yqps47tgze3k753f5ljlj5pzvbpsfb3f5ibq:
    Type:    FlexVolume (a generic volume resource that is
provisioned/attached using an exec based plugin)
    Driver:      Options:  %v


    FSType:     oracle/oci
    SecretRef:  ext4
    ReadOnly:   <nil>
%!(EXTRA bool=false, map[string]string=map[])  default-token-xkljq:
    Type:        Secret (a volume populated by a Secret)
    SecretName:  default-token-xkljq
    Optional:    false
```

```
QoS Class:       BestEffort
Node-Selectors:  <none>
Tolerations:     node.kubernetes.io/not-ready:NoExecute for 300s
                 node.kubernetes.io/unreachable:NoExecute for 300s
Events:
  Type    Reason                 Age   From                  Message
  ----    ------                 ----  ----                  -------
  Normal  Scheduled              50s   default-
scheduler     Successfully assigned flexnginx to kubeworker2
  Normal  SuccessfulMountVolume  49s   kubelet,
kubeworker2  MountVolume.SetUp succeeded for volume "default-token-
xkljq"
  Normal  SuccessfulMountVolume  33s   kubelet,
kubeworker2  MountVolume.SetUp
succeeded for volume "abuwcljrnuyj67r7xgtfryj6yqps47tgze3k753f5ljlj5pz
vbpsfb3f5ibq"
  Normal  Pulling                32s   kubelet, kubeworker2  pulling
image "nginx"
  Normal  Pulled                 30s   kubelet,
kubeworker2  Successfully pulled image "nginx"
  Normal  Created                30s   kubelet, kubeworker2  Created
container
  Normal  Started                30s   kubelet, kubeworker2  Started
container
```

Step 3. On the worker node where pod was created, verify the volume is mounted

```
[root@kubeworker2 ~]# mount | grep
abuwcljrnuyj67r7xgtfryj6yqps47tgze3k753f5ljlj5pzvbpsfb3f5ibq
/dev/sdc on /var/lib/kubelet/plugins/kubernetes.io/flexvolume/oracle/o
ci/mounts/abuwcljrnuyj67r7xgtfryj6yqps47tgze3k753f5ljlj5pzvbpsfb3f5ibq
 type ext4 (rw,relatime,seclabel,stripe=256,data=ordered)
/dev/sdc on /var/lib/kubelet/pods/2c47bc42-3e1c-11e8-be77-
020017004126/volumes/oracle~oci/abuwcljrnuyj67r7xgtfryj6yqps47tgze3k75
3f5ljlj5pzvbpsfb3f5ibq type ext4
(rw,relatime,seclabel,stripe=256,data=ordered)
```

Tutorial

This guide will walk you through creating a Pod with persistent storage. It assumes that you have already installed the Flexvolume driver in your cluster.

See example/nginx.yaml for a finished Kubernetes manifest that ties all these concepts together.

1. Create a block storage volume. This can be done using the `oci` CLI as follows:

```
$ oci bv volume create \
    --availability-domain="aaaa:PHX-AD-1" \
```

```
    --compartment-id
"ocid1.compartment.oc1..aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"
```

1. Add a volume to your `pod.yml` in the format below and named with the last section of your volume's OCID (see limitations). E.g. a volume with the OCID

```
ocid1.volume.oc1.phx.aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
```

Would be named `aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa` in the pod.yml as shown below.

```
volumes:
  - name: "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"
    flexVolume:
      driver: "oracle/oci"
      fsType: "ext4"
```

1. Add volume mount(s) in the appropriate container(s) in your as follows:

```
volumeMounts:
  - name: "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"
    mountPath: /usr/share/nginx/html
```

(Where `"aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"` is the last '.' separated section of the volume OCID.)

Debugging

The Flexvolume driver writes logs to `/usr/libexec/kubernetes/kubelet-plugins/volume/exec/oracle~oci/oci_flexvolume_driver.log` by default.

Assumptions

- If a Flexvolume is specified for a Pod, it will only work with a single replica. (or if there is more than one replica for a Pod, they will all have to run on the same Kubernetes Node). This is because a volume can only be attached to one instance at any one time. Note: This is in common with both the Amazon and Google persistent volume implementations, which also have the same constraint.
- If nodes in the cluster span availability domain you must make sure your Pods are scheduled in the correct availability domain. This can be achieved using the label selectors with the zone/region.

  Using the oci-volume-provisioner makes this much easier.

- For all nodes in the cluster, the instance display name in the OCI API must match with the instance hostname, start with the vnic hostnamelabel or match the public IP. This relies on the requirement that the nodename must be resolvable.

Limitations

Due to kubernetes/kubernetes#44737 ("Flex volumes which implement `getvolumename` API are getting unmounted during run time") we cannot implement `getvolumename`. From the issue:

Detach call uses volume name, so the plugin detach has to work with PV Name

This means that the Persistent Volume (PV) name in the `pod.yml` *must* be the last part of the block volume OCID ('.' separated). Otherwise, we would have no way of determining which volume to detach from which worker node. Even if we were to store state at the time of volume attachment PV names would have to be unique across the cluster which is an unreasonable constraint.

The full OCID cannot be used because the PV name must be shorter than 63 characters and cannot contain '.'s. To reconstruct the OCID we use the region of the master on which `Detach()` is executed so this blocks support for cross region clusters.

---

[i] Kubernetes® is a registered trademark of The Linux Foundation in the United States and other countries, and is used pursuant to a license from The Linux Foundation.